CLAIMS

1. A method comprising:

segmenting a file into multiple blocks;

computing hashes of each of the blocks to produce corresponding block hash values; and

encrypting the blocks using their corresponding block hash values as encryption keys to produce encrypted blocks.

2. A method as recited in claim 1, wherein the segmenting comprises dividing the file into equal size blocks.

3. A method as recited in claim 1, wherein the encrypting comprises encrypting each block using a symmetric cryptographic cipher and the corresponding block hash value as the symmetric encryption key.

4. A method as recited in claim 1, further comprising storing the encrypted blocks as a primary data stream.

5. A method as recited in claim 4, further comprising storing header information in a separate metadata stream.

6. A method as recited in claim 1, further comprising verifying an authenticity of the encrypted blocks independently of one another.

7. A method as recited in claim 1, further comprising modifying content of a block in the file independent of other blocks.

8. A method as recited in claim 1, further comprising constructing an indexing structure to index individual encrypted blocks.

9. A method as recited in claim 8, wherein the constructing comprises creating a leaf node for each corresponding encrypted block, the leaf node containing an access value used to decrypt the corresponding encrypted block and a verification value used to verify the corresponding encrypted block.

10. A method as recited in claim 9, wherein the constructing further comprises hashing an array of the leaf nodes to produce a root.

11. A method as recited in claim 10, wherein the constructing further comprising digitally signing at least the root.

12. A method as recited in claim 10, further comprising:

storing the root together with header information and per user information; and

digitally signing a composite including the root, the header information, and the per user information.

13. A method as recited in claim 9, wherein the constructing further comprises:

grouping leaf nodes into multiple groups;

hashing each group of leaf nodes to form intermediate nodes; and

hashing an array of the intermediate nodes to produce a root.

14. A method as recited in claim 13, wherein the constructing further comprises digitally signing at least the root.

15. A method as recited in claim 1, further comprising digitally signing at least a portion of the file.

16. A method as recited in claim 1, further comprising generating a delegation certificate that grants other entities permission to collectively authenticate the file in absence of the signature of a last writer to the file.

17. A method as recited in claim 1, wherein the file comprises a sparse file.

18. A method as recited in claim 1, further comprising:

encrypting the block hash values with one or more access keys; and

encrypting the one or more access keys using one or more keys of users who are granted access to the file.

19. A method as recited in claim 18, wherein the one or more keys of users comprise one or more public keys.

20. A data structure, embodied on a computer-readable medium, produced by the method of claim 1.

21. One or more computer readable media comprising computer-executable instructions that, when executed, perform the method as recited in claim 1.

22. A method comprising:

segmenting a file into multiple blocks;

computing hashes of each of the blocks to produce corresponding block hash values;

encrypting the blocks using their corresponding block hash values as encryption keys to produce encrypted blocks;

storing the encrypted blocks as a primary data stream;

creating an indexing structure to index individual encrypted blocks, the indexing structure containing a leaf node for each corresponding encrypted block, the leaf node containing an access value formed by encrypting the block hash value for the corresponding encrypted block using an access key and a verification value formed by hashing the corresponding encrypted block;

storing the indexing structure in a separate metadata stream; and

encrypting the access key using a public key of a user who is granted access to the file.

23. A method as recited in claim 22, wherein the segmenting comprises dividing the file into equal size blocks.

24. A method as recited in claim 22, wherein the encrypting of the blocks comprises encrypting each block using a symmetric cryptographic cipher and the corresponding block hash value as the symmetric encryption key.

25. A method as recited in claim 22, further comprising verifying an authenticity of a target encrypted block independently of other encrypted blocks by traversing the indexing structure to a leaf node associated with the target encrypted block and using the verification value in the leaf node associated with the target encrypted block.

26. A method as recited in claim 22, further comprising:

traversing the indexing structure to a leaf node associated with a target block;

decrypting the target block using the access value of the leaf node associated with the target block; and

reading the target block following said decrypting.

27. A method as recited in claim 26, further comprising:

modifying the target block of the file to produce a modified target block;

computing a hash value of the modified target block;

encrypting the modified target block using the hash value as an encryption key to produce a modified encrypted block; and

recreating a new leaf node for the modified encrypted block.

28. A method as recited in claim 22, wherein the creating further comprises:

grouping leaf nodes into multiple groups;

hashing each group of leaf nodes to form intermediate nodes of the indexing structure; and

hashing an array of the intermediate nodes to produce a root.

29. A method as recited in claim 28, wherein the constructing further comprises digitally signing at least the root.

30. A method as recited in claim 22, further comprising digitally signing at least a portion of the metadata stream.

31. A method as recited in claim 22, further comprising generating a delegation certificate that grants other entities permission to collectively authenticate the file in absence of the signature of a last writer to the file.

32. A method as recited in claim 22, wherein the file comprises a sparse file in which at least one of the blocks contains no data, the method further comprising:

differentiating non-data blocks of the sparse file that contain no substantive content from the data blocks of the sparse file that contain substantive data; and

deallocating portions of the metadata stream that pertain to the non-data blocks in the data stream.

33. A data structure, embodied on a computer-readable medium, produced by the method of claim 22.

34. One or more computer readable media comprising computer-executable instructions that, when executed, perform the method as recited in claim 22.

35. A method comprising:

creating a primary data stream containing an encrypted file that is encrypted using at least one hash of some contents of the file; and

creating a metadata stream containing information pertaining to the encrypted file, the information including decryption capabilities used to decrypt the file and verification capabilities used to verify portions of the file without access to decryption keys.

36. A method comprising:

accessing a file composed of a data stream and a metadata stream, the data stream containing multiple encrypted blocks that are each encrypted using hashes of a plaintext version of the encrypted blocks, the metadata stream containing an indexing structure to index to the individual encrypted blocks, the indexing structure having a leaf node for each corresponding encrypted block that contains a verification value used to verify the corresponding encrypted block;

traversing the indexing structure to a leaf node associated with a target encrypted block; and

verifying an authenticity of the target encrypted block independently of other encrypted blocks by using the verification value in the leaf node associated with the target encrypted block.

37. A method as recited in claim 36, wherein the indexing structure contains a root and zero or more intervening nodes between the root and the leaf nodes, the traversing further comprising verifying an authenticity of the root and any intervening nodes on a path from the root to the leaf node associated with the target encrypted block.

38. A method as recited in claim 36, wherein the indexing structure is at least partially digitally signed with a digital signature, the method further comprising evaluating an authenticity of the digital signature.

39. A method as recited in claim 36, wherein the verification value in the leaf node is a hash value of the target encrypted block, and the verifying comprises:

computing a current hash value of the target encrypted block; and

comparing the current hash value with the hash value in the leaf node.

40. A method for reading a file stored in a distributed file system, the file containing a data stream with multiple encrypted blocks and a metadata stream with an indexing structure to index the encrypted blocks individually, the indexing structure having a leaf node for each corresponding encrypted block that contains an access value used to decrypt the corresponding encrypted block, the method comprising:

indexing into the indexing structure to a leaf node associated with a target encrypted block;

decrypting the target encrypted block using the access value of the leaf node associated with the target encrypted block; and

reading the target encrypted block following said decrypting.

41. A method as recited in claim 40, wherein the access value in the leaf node is an encrypted version of symmetric key used to encrypt the file block, the symmetric key being generated by hashing the file block.

42. A method for writing to a file stored in a distributed file system, the file containing a data stream with multiple encrypted blocks and a metadata stream with an indexing structure to index to the encrypted blocks individually, the method comprising:

modifying a block of the file;

computing a hash value of the block;

encrypting the block using the hash value as an encryption key to produce an encrypted block; and

reconstructing a portion of the indexing structure that references the encrypted block.

43. A method as recited in claim 42, wherein the modifying the block comprises writing data to the block.

44. A method as recited in claim 42, wherein the indexing structure includes a leaf node for each corresponding encrypted block, and the reconstructing comprises creating a new leaf node for the encrypted block, the new leaf node containing an encrypted version of the hash value and a hash of the encrypted block.

45. A method comprising:

segmenting a sparse file into multiple blocks;

differentiating non-data blocks in the sparse file that contain no substantive content from data blocks in the sparse file that contain substantive data;

creating an indexing structure to index individual blocks; and

deallocating storage of the non-data blocks and portions of the indexing structure that reference the non-data blocks.

46. A method as recited in claim 45, further comprising storing the data blocks in a data stream and the indexing structure in a metadata stream.

47. A method as recited in claim 45, further comprising:

computing a hash of each of the data blocks to produce block hash values; and

encrypting the data blocks using their corresponding block hash values as encryption keys to produce encrypted data blocks.

48. A method as recited in claim 47, wherein the indexing structure contains first leaf nodes for each corresponding data blocks and second leaf nodes for each corresponding non-data block, the first leaf nodes containing an access value for use in decrypting the encrypted data blocks and a verification value for use in verifying individual encrypted data block independently of other encrypted data blocks.

49. A data structure, embodied on a computer-readable medium, produced by the method of claim 45.

50. One or more computer readable media comprising computer-executable instructions that, when executed, perform the method as recited in claim 45.

51. A method comprising:

segmenting a sparse file into multiple blocks, the sparse file containing at least one non-data block that contains no substantive data;

differentiating the non-data blocks from data blocks of the sparse file that contain substantive data;

computing hashes of each of the data blocks to produce block hash values;

encrypting the data blocks using their corresponding block hash values as encryption keys to produce encrypted data blocks;

creating an indexing structure to index individual blocks, the indexing structure containing first leaf nodes for each corresponding encrypted data block and second leaf nodes for each corresponding non-data block, the first leaf nodes containing an access value formed by encrypting the block hash value for the corresponding encrypted block using an access key and a verification value formed by hashing the corresponding encrypted block; and

setting the second leaf nodes to a first binary value.

52. A method as recited in claim 51, further comprising storing the encrypted blocks in a data stream and the indexing structure in a metadata stream.

53. A method as recited in claim 52, further comprising deallocating portions of the metadata stream that hold the second leaf nodes.

54. A data structure, embodied on a computer-readable medium, produced by the method of claim 51.

55. One or more computer readable media comprising computer-executable instructions that, when executed, perform the method as recited in claim 51.

56. One or more computer readable media comprising computer-executable instructions that, when executed, direct a computing device to:

segment a file into multiple blocks;

hash each of the blocks to produce block hash values;

encrypt the blocks using their corresponding block hash values as encryption keys to produce encrypted blocks;

create an indexing structure to index individual encrypted blocks, the indexing structure containing a leaf node for each corresponding encrypted block, the leaf node containing an access value formed by encrypting the block hash value for the corresponding encrypted block using an access key and a verification value formed by hashing the corresponding encrypted block;

encrypt the access key using a public key of a user who is granted access to the file.

57. One or more computer readable media as recited in claim 56, further comprising computer-executable instructions that, when executed, direct a computing device to:

store the encrypted blocks as a primary data stream; and

store the indexing structure in a separate metadata stream.

58. One or more computer readable media as recited in claim 56, further comprising computer-executable instructions that, when executed, direct a computing device to segment the file into equal size blocks.

59. One or more computer readable media as recited in claim 56, wherein the blocks are encrypted using a symmetric cryptographic cipher and the access key is encrypted using an asymmetric cryptographic cipher.

60. One or more computer readable media as recited in claim 56, further comprising computer-executable instructions that, when executed, direct a computing device to verify an authenticity of a target encrypted block independently of other encrypted blocks by traversing the indexing structure to a leaf node associated with the target encrypted block and using the verification value in the leaf node associated with the target encrypted block.

61. A method as recited in claim 60, wherein the indexing structure contains a root and zero or more intervening nodes between the root and the leaf nodes, the traversing further comprising verifying an authenticity of the root and any intervening nodes on a path from the root to the leaf node associated with the target encrypted block.

62. One or more computer readable media as recited in claim 56, further comprising computer-executable instructions that, when executed, direct a computing device to:

decrypt a target block using an access value of a leaf node associated with the target block; and

read the target block after it is decrypted.

63. One or more computer readable media as recited in claim 62, further comprising computer-executable instructions that, when executed, direct a computing device to:

modify the target block to produce a modified target block;

hash the modified target block to produce a hash value;

encrypt the modified target block using the hash value as an encryption key to produce a modified encrypted block; and

recreate a new leaf node for the modified encrypted block.

64. One or more computer readable media as recited in claim 56, further comprising computer-executable instructions that, when executed, direct a computing device to:

group leaf nodes into multiple groups;

hash each group of leaf nodes to form intermediate nodes of the indexing structure; and

hash an array of the intermediate nodes to produce a root.

65. One or more computer readable media as recited in claim 64, further comprising computer-executable instructions that, when executed, direct a computing device to digitally sign at least the root.

66. One or more computer readable media as recited in claim 56, further comprising computer-executable instructions that, when executed, direct a computing device to digitally sign at least a portion of the metadata stream.

67. One or more computer readable media as recited in claim 56, further comprising computer-executable instructions that, when executed, direct a computing device to generate a delegation certificate that grants other entities permission to collectively authenticate the file in absence of the signature of a last writer to the file.

68. One or more computer readable media as recited in claim 56, wherein the file comprises a sparse file in which at least one of the blocks contains no substantive data, the media further comprising computer-executable instructions that, when executed, direct a computing device to:

differentiate non-data blocks of the sparse file that contain no substantive content from the data blocks of the sparse file that contain substantive data; and

deallocate portions of the metadata stream that pertain to the non-data blocks in the data stream.

69. In a distributed file system that stores files across multiple computers, wherein each file contains a data stream with multiple encrypted blocks and a metadata stream with an indexing structure to index the encrypted blocks individually, the indexing structure having a leaf node for each corresponding encrypted block that contains a verification value used to verify the corresponding encrypted block, one or more computer readable media comprising computer-executable instructions that, when executed, direct a computing device to:

traverse the indexing structure to a leaf node associated with a target encrypted block; and

verify an authenticity of the target encrypted block independently of other encrypted blocks by using the verification value in the leaf node associated with the target encrypted block.

70. One or more computer readable media as recited in claim 69, wherein the indexing structure contains a root and zero or more intervening nodes between the root and the leaf nodes, further comprising computer-executable instructions that, when executed, direct a computing device to verify an authenticity of the root and any intervening nodes on a path from the root to the leaf node associated with the target encrypted block.

71. In a distributed file system that stores files across multiple computers, wherein each file contains a data stream with multiple encrypted blocks and a metadata stream with an indexing structure to index the encrypted blocks individually, the indexing structure having a leaf node for each corresponding encrypted block that contains an access value used to decrypt the corresponding encrypted block, one or more computer readable media comprising computer-executable instructions that, when executed, direct a computing device to:

index into the indexing structure to a leaf node associated with a target encrypted block;

decrypt the target encrypted block using the access value of the leaf node associated with the target encrypted block; and

read the target encrypted block following said decrypting.

72. In a distributed file system that stores files across multiple computers, the file containing a data stream with multiple encrypted blocks and a metadata stream with an indexing structure to index to the encrypted blocks individually, one or more computer readable media comprising computer-executable instructions that, when executed, direct a computing device to:

modify a block of the file;

compute a hash value of the block;

encrypt the block using the hash value as an encryption key to produce an encrypted block; and

reconstruct a portion of the indexing structure that references the encrypted block.

73. One or more computer readable media comprising computer-executable instructions that, when executed, direct a computing device to:

segment a sparse file into multiple blocks, the sparse file containing at least one non-data block that contains no substantive data;

differentiate the non-data blocks from data blocks of the sparse file that contain substantive data;

compute hashes of each of the data blocks to produce block hash values;

encrypt the data blocks using their corresponding block hash values as encryption keys to produce encrypted data blocks;

creating an indexing structure to index the non-data blocks and the encrypted data blocks; and

deallocate portions of the indexing structure that reference the non-data blocks.

74. A distributed file system comprising:

a client component resident at a first computer to facilitate creation of a file by segmenting the file into multiple blocks and encrypting each block using its own hash value as an encryption key; and

a server component resident at a second computer to store the encrypted file.

75. A distributed file system as recited in claim 74, wherein the client component divides the file into equal size blocks.

76. A distributed file system as recited in claim 74, wherein the encrypted blocks are stored as a primary data stream.

77. A distributed file system as recited in claim 76, wherein the client component creates header information that is stored as a separate metadata stream.

78. A distributed file system as recited in claim 74, wherein the client component verifies an authenticity of the encrypted blocks independently of one another.

79. A distributed file system as recited in claim 74, wherein the client component modifies content of a block in the file independent of other blocks.

80. A distributed file system as recited in claim 74, wherein the client component digitally signs the file.

81. A component in a distributed file system in which file are stored across multiple distributed computers, the component comprising:

a segmenting module to divide a file into multiple blocks;

a hash module to hash each of the blocks to produce block hash values;

a cryptographic engine to encrypt the blocks using their corresponding block hash values as encryption keys to produce encrypted blocks; and

an index builder to create an indexing structure for indexing individual encrypted blocks, the indexing structure containing a leaf node for each corresponding encrypted block, the leaf node containing an access value formed by encrypting the block hash value for the corresponding encrypted block using an access key and a verification value formed by hashing the corresponding encrypted block.

82. A component as recited in claim 81, wherein the cryptographic engine is further configured to encrypt the access key using a key of a user who is granted access to the file.

83. A component as recited in claim 81, wherein the segmenting module divides the file into equal size blocks.

84. A component as recited in claim 81, wherein cryptographic engine employs a symmetric cryptographic cipher to encrypt the blocks.

85. A component as recited in claim 81, further comprising a verification module to verify an authenticity of a target encrypted block independently of other encrypted blocks by traversing the indexing structure to a leaf node associated with the target encrypted block and using the verification value in the leaf node associated with the target encrypted block.

86. A component as recited in claim 85, wherein the indexing structure contains a root and zero or more intervening nodes between the root and the leaf nodes, the verification module being configured to verify an authenticity of the root and any intervening nodes on a path from the root to the leaf node associated with the target encrypted block.

87. A component as recited in claim 81, further comprising a control module to index into the indexing structure to a leaf node associated with a target block, decrypt the target block using the access value of the leaf node associated with the target block, and read the target block.

88. A component as recited in claim 87, where upon modification of the target block:

the hash module hashes the modified target block to produce a new hash value;

the cryptographic engine encrypts the modified target block using the new hash value as an encryption key to produce a modified encrypted block; and

the index builder creates a new leaf node for the modified encrypted block.

89. A component as recited in claim 81, wherein the index builder is configured to create intermediate nodes that index the leaf nodes.

90. A component as recited in claim 81, further comprising a signing module to digitally sign at least a portion of the indexing structure.

91. A component in a distributed file system in which files are stored across multiple distributed computers, the component comprising:

a segmenting module to divide a sparse file into multiple blocks, the sparse file containing at least one non-data block that contains no substantive data;

a control module to differentiate the non-data blocks from data blocks of the sparse file that contain substantive data;

a hash module to hash each of the data blocks to produce block hash values;

a cryptographic engine to encrypt the data blocks using their corresponding block hash values as encryption keys to produce encrypted blocks; and

an index builder to create an indexing structure to index individual blocks, the indexing structure containing first leaf nodes for each corresponding encrypted block and second leaf nodes for each corresponding non-data block, the first leaf nodes containing an access value formed by encrypting the block hash value for the corresponding encrypted block using an access key and a verification value

formed by hashing the corresponding encrypted block, the second leaf nodes being set to a first binary value.

92. A component as recited in claim 91, wherein the encrypted blocks are stored in a data stream and the indexing structure is stored in a metadata stream.

93. A component as recited in claim 92, wherein the control module deallocates portions of the metadata stream that hold the second leaf nodes.

94. A distributed file system comprising:

means for creating a primary data stream containing an encrypted file that is encrypted using at least one hash of some contents of the file; and

means for creating a metadata stream containing information pertaining to the encrypted file, the information including decryption capabilities used to decrypt the file and verification capabilities used to verify the file.

95. A distributed file system as recited in claim 94, wherein the file in the primary stream comprises multiple encrypted blocks, where each block is hashed and encrypted using a hash value of the block as an encryption key.

96. A distributed file system as recited in claim 94, wherein the means for creating the metadata stream comprises means for constructing leaf nodes for corresponding blocks, the leaf nodes containing an access value used to decrypt the corresponding encrypted block and a verification value used to verify the corresponding encrypted block.

97. A data structure stored on a computer-readable medium, comprising:

a primary data stream containing an encrypted file composed of multiple encrypted blocks, each block being separately encrypted by a symmetric cipher that uses a hash of the block as an encryption key;

a metadata stream containing information pertaining to the encrypted file.

98. A data structure as recited in claim 97, wherein information includes decryption capabilities used to decrypt the file.

99. A data structure as recited in claim 97, wherein information includes verification capabilities used to verify the file.

100. A data structure as recited in claim 97, wherein the metadata stream comprises:

a header with one or more fields that describe the encrypted file; and

an indexing structure to index individual encrypted blocks.

101. A data structure stored on a computer-readable medium, comprising:

multiple encrypted file blocks, each encrypted file block being encrypted by a symmetric cipher that uses a hash of the block as an encryption key; and

an indexing structure to index individual encrypted file blocks independently of other encrypted file blocks.

102. A data structure as recited in claim 101, wherein the indexing structure comprises a leaf node for each corresponding encrypted block, the leaf node containing an access value formed by encrypting the hash of the block using a randomly generated key and a verification value formed by hashing the corresponding encrypted block.

103. A data structure as recited in claim 102, further comprising a user key list containing one or more identities of user who have access to the encrypted file blocks, each identity including an entry with an encrypted version of the randomly generated key that is encrypted using the user's public key.

104. A data structure as recited in claim 101, wherein the indexing structure comprises:

a leaf node for each corresponding encrypted block, the leaf node containing an access value formed by encrypting the hash of the block using a randomly generated key and a verification value formed by hashing the corresponding encrypted block; and

a root node formed by hashing an array of the leaf nodes.

105. A data structure as recited in claim 104, wherein the indexing structure further comprises a digital signature produced by digitally signing at least the root node.